Prof. Dr. Karsten Urban
Dipl.-Math. oec. Klaus Stolle
Institut für Numerische Mathematik
Universität Ulm

WiSe 2015/2016
Blatt 3
30.10.2015

Angewandte Numerik 2

Abgabetermin: Freitag, 06.11.2015, **vor der Übung**

Für dieses Übungsblatt gibt es 22 Theorie- und 23 Matlab-Punkte, sowie 10 Theorie- und 6 Matlab-Zusatzpunkte. Punkte, die mit einem * gekennzeichnet sind, sind Zusatzpunkte.
Die 50-Prozent-Grenzen liegen aktuell (inklusive Blatt 3) bei 35,5 Theoriepunkten und 27,5 Matlabpunkten.

Aufgabe 11 *(Programmieraufgabe: Effiziente Auswertung von B-Spline-Kurven)*
(6T+6M+1M+1M+2T*+2M*+4M* Punkte)

In Aufgabe 9 vom letzten Übungsblatt 2 haben wir die folgenden Messwerte

x_i	0	1	2	3
y_i	2	3	3	5

durch einen quadratischen B-Spline s interpoliert. Mit der zusätzlichen Randbedingung $s'(0) = 0$ haben wir die Koeffizienten $c_0 = 2$, $c_1 = 2$, $c_2 = 4$, $c_3 = 2$ und $c_4 = 5$ zu den B-Spline-Basisfunktionen $N_j^2(t)$, $j = 0, \ldots, 4$ vom Grad 2 erhalten. Damit gilt für s die Darstellung

$$s(t) = 2N_0^2(t) + 2N_1^2(t) + 4N_2^2(t) + 2N_3^2(t) + 5N_4^2(t).$$

a) Werten Sie den B-Spline s an der Stelle $x = \frac{1}{2}$ effizient von Hand aus.
Hinweise: Verwenden Sie die Knotenfolge $\mathcal{T} = \{t_0, \ldots, t_7\}$ aus Aufgabe 9. Zur effizienten Auswertung brauchen Sie die B-Spline-Basisfunktionen $N_j^2(t)$ nicht zu kennen.

b) Schreiben Sie eine Matlabfunktion `sxVektor = bSplineAuswertung(ti, grad, cj, xVektor)`, die einen durch die Knotenfolge `ti` $= (t_0, \ldots, t_{n+1})$ und den Koeffizientenvektor `cj` $= (c_0, \ldots, c_{n+1-\texttt{grad}})$ gegebenen B-Spline vom Grad `grad` an den Stellen `xVektor` $= (x_0, \ldots, x_m)$ auswertet. Der Rückgabewert `sxVektor` soll dabei die zu den Werten x_i in `xVektor` gehörenden Werte $s(x_i)$ des B-Splines enthalten. Achten Sie auf eine effiziente Implementierung.

c) Schreiben Sie ein Matlabskript `mainBSplineAuswertung`, mit dem Sie den B-Spline s an der Stelle $x = \frac{1}{2}$ effizient auswerten. Vergleichen Sie das Ergebnis mit Ihrem Ergebnis aus dem ersten Aufgabenteil.

d) Plotten Sie den B-Spline aus Aufgabe 9.

e) Überlegen Sie sich, wie Sie die B-Spline-Basisfunktionen $N_j^k(t)$ vom Grad k mit Hilfe Ihrer Matlabfunktion `bSplineAuswertung` auswerten und plotten können.

f) Plotten Sie alle stückweise konstanten B-Spline-Basisfunktion vom Grad 0 zur obigen Knotenfolge in ein Schaubild. Verwenden Sie für die einzelnen Basisfunktionen verschiedene Farben.

g) Plotten Sie auch die B-Spline-Basisfunktion vom Grad 1, vom Grad 2 und vom Grad 3 (zur obigen Knotenfolge) jeweils in ein Schaubild. Achten Sie auf eine saubere Beschriftung Ihrer Schaubilder.

Aufgabe 12 *(Jakobi-Verfahren und Gauß-Seidel-Verfahren)* (8T*+4T+8T+4T Punkte)

a) Führen Sie von Hand für das lineare Gleichungssystem

$$\begin{pmatrix} 4 & 1 & 1 \\ 1 & 4 & 1 \\ 0 & 1 & 2 \end{pmatrix} x = \begin{pmatrix} 6 \\ 6 \\ 3 \end{pmatrix}$$

jeweils zwei Schritte des Jakobi- und des Gauß-Seidel-Verfahrens durch, ohne dabei Inverse von Matrizen zu berechnen. Wählen Sie als Startvektor den Vektor $x^{(0)} = (0, 0, 0)^T$ und geben Sie alle Ihre Rechenschritte an.

b) Können Sie für das obige Beispiel Aussagen machen, ob die beiden Verfahren konvergieren?

c) Leiten Sie für das Jakobi- und das Gauß-Seidel-Verfahren jeweils eine komponentenweise Darstellung der Iterationsvorschrift her.

d) Moderne Computer, die für Simulationen eingesetzt werden, haben mehrere Prozessoren, die wiederum jeweils mehrere Prozessorkerne haben. Bei der Parallelisierung versucht man, Berechnungen wie beispielsweise einen Iterationsschritt eines Iterations-Verfahrens auf mehrere dieser Prozessorkerne so aufzuteilen, dass die einzelnen Arbeitsschritte dort gleichzeitig und möglichst unabhängig voneinander durchgeführt werden können.
Diskutieren Sie: Welches der beiden Verfahren lässt sich besser parallelisieren?

Aufgabe 13 *(Programmieraufgabe: Jacobi- und Gauß-Seidel-Verfahren)* (15M Punkte)

In dieser Aufgabe sollen sie das Jacobi- und das Gauss-Seidel-Verfahren vergleichen. Schreiben Sie dazu ein Matlabskript `vergleichJacobiGaussSeidel` in dem Sie für eine zufällige, diagonaldominante 1000×1000 - Matrix A und einen zufälligen 1000×1 - Vektor b das Gleichungssystem $Ax = b$ mit beiden Verfahren lösen. Gehen Sie dabei wie folgt vor.

(1) Initialisieren Sie die Matrix A, den Vektor b und den Startvektor $x0$, der nur Nullen enthält.
Achtung: A soll zufällig und diagonaldominant sein.

(2) Berechnen Sie die exakte Lösung x^* mit dem Backslash-Operator `'\'`.

(3) Führen Sie jeweils 12 Schritte des Jacobi- und des Gauß-Seidel-Verfahrens durch. Berechnen Sie für jeden Schritt den Fehler $||x^{(k)} - x^*||_2$. (Sie dürfen für die Norm die Matlab-Funktion `norm` verwenden.)

(4) Plotten Sie beide Fehler über k in eine Grafik. Wählen Sie die Skalierung der Achsen so, dass näherungsweise Geraden entstehen.

Achten Sie darauf, dass ihre Iterationen effizient ausgeführt werden. Insbesondere ist es effizienter, die Iterationsmatrizen für die beiden Verfahren vor der Iteration nur einmal zu berechnen und nicht in jedem Iterationsschritt neu.
Was lässt sich über das Konvergenzverhalten der beiden Verfahren sagen?

Hinweise:
Die Programmieraufgaben sind in Matlab zu erstellen. Senden Sie alle Matlab-Files in einer E-mail mit dem Betreff **Loesung-Blatt03** an **angewandte.numerik@uni-ulm.de** (Abgabetermin jeweils wie beim Theorieteil). Drucken Sie zusätzlich allen Programmcode sowie die Ergebnisse aus und geben Sie diese vor der Übung ab. Der Source Code sollte strukturiert und, wenn nötig, dokumentiert sein.